

Project – Big Data And Spark

Name : Keerthi Rajan

Market Analysis in Banking Domain

DESCRIPTION

Background and Objective:

Your client, a Portuguese banking institution, ran a marketing campaign to convince potential customers to invest in a bank term deposit scheme.

The marketing campaigns were based on phone calls. Often, the same customer was contacted more than once through phone, in order to assess if they would want to subscribe to the bank term deposit or not. You have to perform the marketing analysis of the data generated by this campaign.

Domain: Banking (Market Analysis)

Analysis tasks to be done:-

- The data size is huge and the marketing team has asked you to perform the below analysis-
- Load data and create a Spark data frame
- Give the maximum, mean, and minimum age of the average targeted customer
- Check the quality of customers by checking average balance, median balance of customers
- Check if age matters in marketing subscription for deposit
- Check if marital status mattered for a subscription to deposit
- Check if age and marital status together mattered for a subscription to deposit scheme
- Do feature engineering for the bank and find the right age effect on the campaign.

Code and Outputs

Dataset

Uploaded json file

The screenshot shows the Hue File Browser interface. The browser is displaying a file named 'bank_edited.json' located at the path '/ user / rajrkeerthi_gmail / bank_edited.json'. The file is a JSON document containing a single record with the following fields: 'age' (58), 'job' ('management'), 'marital' ('married'), 'education' ('tertiary'), 'default' ('no'), 'balance' (2143), 'housing' ('yes'), 'loan' ('no'), 'contact' ('unknown'), 'day' (5), 'month' ('may'), 'duration' (261), 'campaign' (1), 'pdays' (-1), 'previous' (0), and 'poutcome' ('unknown'). The interface also shows a sidebar with a file list containing 'bank_edited.json' and a search bar. The top navigation bar includes the Hue logo and a search bar.

Web Console

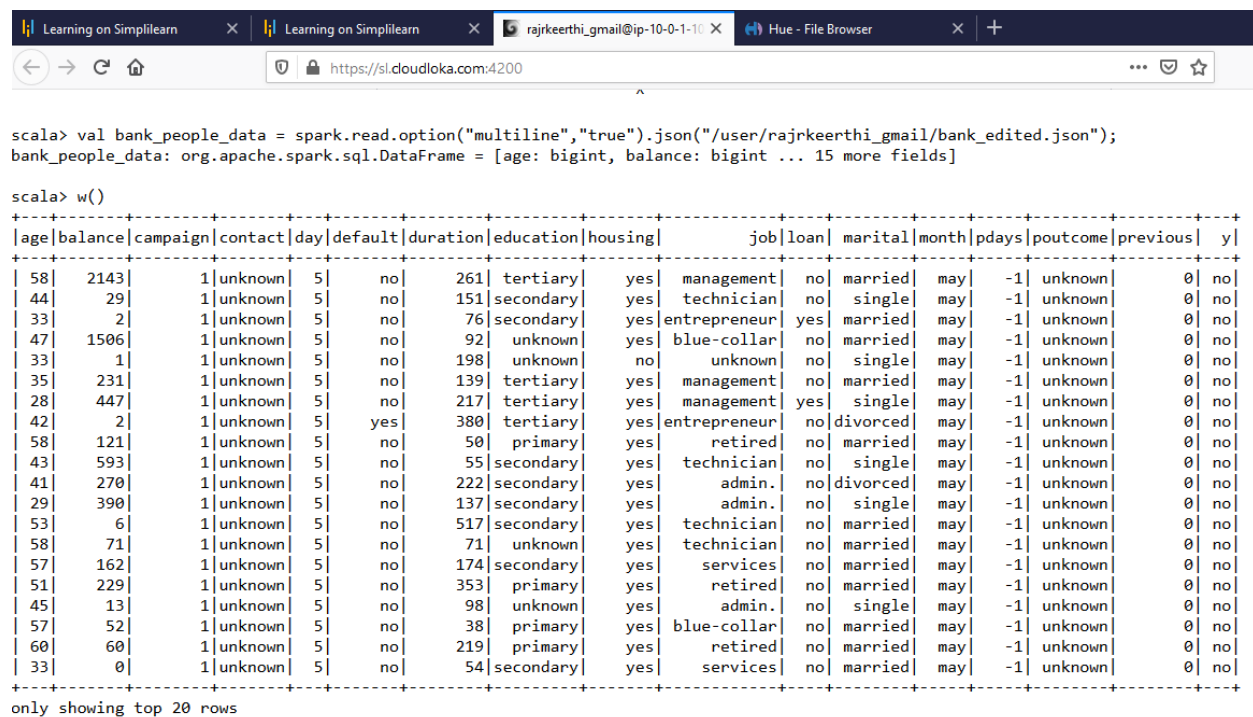
The screenshot shows the Hue Web Console interface. The console displays the output of a Spark2 shell session. The output includes the login information for the user 'rajrkeerthi_gmail' on the host 'ip-10-0-1-10'. The session shows the user entering the command 'spark2-shell' and the resulting Spark context 'Web UI available at http://ip-10-0-1-10.ec2.internal:42002'. The console also displays the Spark session ID 'application_1588139849525_0120' and the Spark session name 'spark'. The console output ends with the text 'Welcome to' followed by a decorative ASCII art logo and the text 'version 2.4.0.cloudera2'. Below the logo, the console displays the Scala version '2.11.12 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_144)' and the prompt 'scala>'.

Code

```
val
bank_people_data=spark.read.option("multiline","true").json("/user/rajrkeerthi_gmail/bank_edited.json");

bank_people_data.show()
```

Output



```
scala> val bank_people_data = spark.read.option("multiline","true").json("/user/rajrkeerthi_gmail/bank_edited.json");
bank_people_data: org.apache.spark.sql.DataFrame = [age: bigint, balance: bigint ... 15 more fields]

scala> w()
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|age|balance|campaign|contact|day|default|duration|education|housing|job|loan|marital|month|pdays|poutcome|previous|y|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|58|2143|1|unknown|5|no|261|tertiary|yes|management|no|married|may|-1|unknown|0|no|
|44|29|1|unknown|5|no|151|secondary|yes|technician|no|single|may|-1|unknown|0|no|
|33|2|1|unknown|5|no|76|secondary|yes|entrepreneur|yes|married|may|-1|unknown|0|no|
|47|1506|1|unknown|5|no|92|unknown|yes|blue-collar|no|married|may|-1|unknown|0|no|
|33|1|1|unknown|5|no|198|unknown|no|unknown|no|single|may|-1|unknown|0|no|
|35|231|1|unknown|5|no|139|tertiary|yes|management|no|married|may|-1|unknown|0|no|
|28|447|1|unknown|5|no|217|tertiary|yes|management|yes|single|may|-1|unknown|0|no|
|42|2|1|unknown|5|yes|380|tertiary|yes|entrepreneur|no|divorced|may|-1|unknown|0|no|
|58|121|1|unknown|5|no|50|primary|yes|retired|no|married|may|-1|unknown|0|no|
|43|593|1|unknown|5|no|55|secondary|yes|technician|no|single|may|-1|unknown|0|no|
|41|270|1|unknown|5|no|222|secondary|yes|admin.|no|divorced|may|-1|unknown|0|no|
|29|390|1|unknown|5|no|137|secondary|yes|admin.|no|single|may|-1|unknown|0|no|
|53|6|1|unknown|5|no|517|secondary|yes|technician|no|married|may|-1|unknown|0|no|
|58|71|1|unknown|5|no|71|unknown|yes|technician|no|married|may|-1|unknown|0|no|
|57|162|1|unknown|5|no|174|secondary|yes|services|no|married|may|-1|unknown|0|no|
|51|229|1|unknown|5|no|353|primary|yes|retired|no|married|may|-1|unknown|0|no|
|45|13|1|unknown|5|no|98|unknown|yes|admin.|no|single|may|-1|unknown|0|no|
|57|52|1|unknown|5|no|38|primary|yes|blue-collar|no|married|may|-1|unknown|0|no|
|60|60|1|unknown|5|no|219|primary|yes|retired|no|married|may|-1|unknown|0|no|
|33|0|1|unknown|5|no|54|secondary|yes|services|no|married|may|-1|unknown|0|no|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 20 rows
```

Code

```
bank_people_data.select(max($"age")).show()

bank_people_data.select(min($"age")).show()

bank_people_data.select(avg($"age")).show()

bank_people_data.select(avg($"balance")).show()
```

Output

```
scala> bank_people_data.select(max($"age")).show()
+-----+
|max(age)|
+-----+
|      95|
+-----+

scala> bank_people_data.select(min($"age")).show()
+-----+
|min(age)|
+-----+
|      18|
+-----+

scala> bank_people_data.select(avg($"age")).show()
+-----+
|      avg(age)|
+-----+
|40.93621021432837|
+-----+

scala> bank_people_data.select(avg($"balance")).show()
+-----+
|      avg(balance)|
+-----+
|1362.2720576850766|
+-----+
```

Code

```
val median = spark.sql("SELECT percentile_approx(balance, 0.5) FROM datanewtable").show()
```

Output

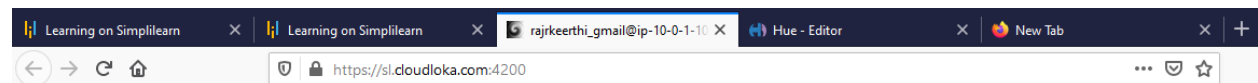
[illegible]

Code

```
val agedata = spark.sql("select age, count(*) as number from datanewtable where y='yes' group by age order by number desc")
```

```
agedata.show()
```

Output



The screenshot shows a web browser with multiple tabs. The active tab is titled "rajkeerthi_gmail@ip-10-0-1-10" and displays the Hue Editor interface. The URL bar shows "https://sl.cloudloka.com:4200". The editor contains the following Scala code:

```
scala> val agedata = spark.sql("select age, count(*) as number from datanewtable where y='yes' group by age order by number desc")
agedata: org.apache.spark.sql.DataFrame = [age: bigint, number: bigint]
```

The output of the `agedata.show()` command is displayed below the code:

```
scala> agedata.show()
+-----+
|age|number|
+-----+
| 32|  221|
| 30|  217|
| 33|  210|
| 35|  209|
| 31|  206|
| 34|  198|
| 36|  195|
| 29|  171|
| 37|  170|
| 28|  162|
| 38|  144|
| 39|  143|
| 27|  141|
| 26|  134|
| 41|  120|
| 46|  118|
| 40|  116|
| 25|  113|
| 47|  113|
| 42|  111|
+-----+
only showing top 20 rows
```

Code

```
val maritaldata = spark.sql("select marital, count(*) as number from datanewtable where y='yes' group by marital order by number desc")
```

```
maritaldata.show()
```

Output

```
scala> val maritaldata = spark.sql("select marital, count(*) as number from datanewtable where y='yes' group by marital order by number desc")
maritaldata: org.apache.spark.sql.DataFrame = [marital: string, number: bigint]

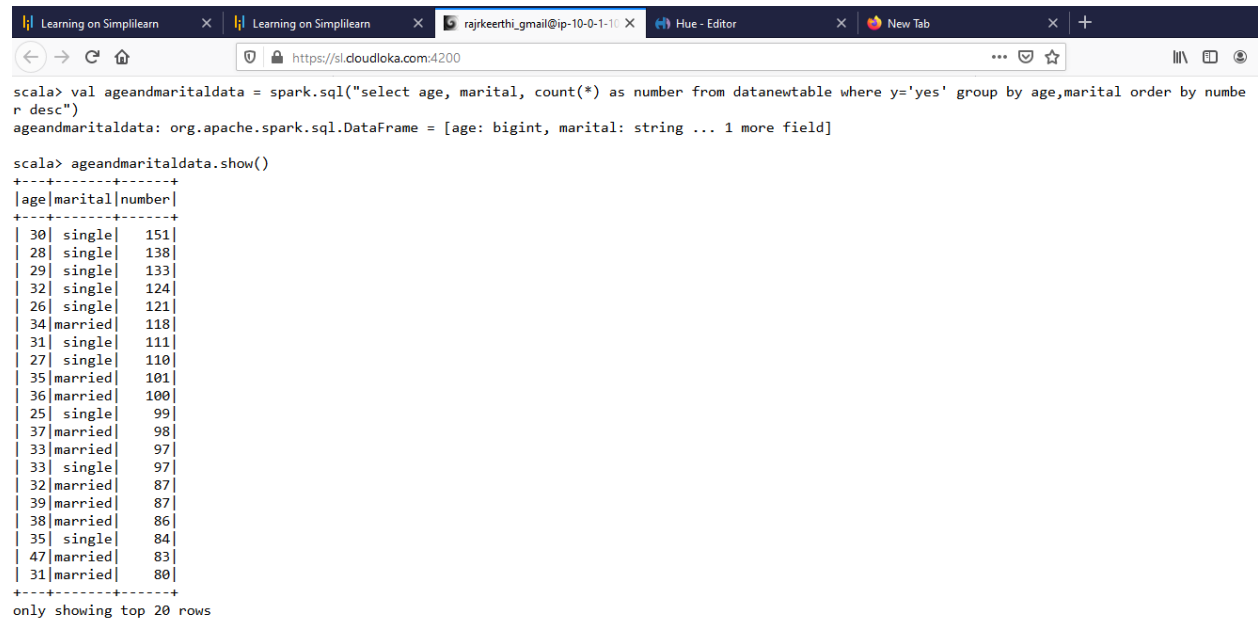
scala> maritaldata.show()
+-----+
|marital|number|
+-----+
|married|  2755|
|single|  1912|
|divorced|  622|
+-----+
```

Code

```
val ageandmaritaldata = spark.sql("select age, marital, count(*) as number from datanewtable where y='yes' group by age,marital order by number desc")

ageandmaritaldata.show()
```

Output



The screenshot shows a web browser window with the URL `https://sl.cloudloka.com:4200`. The interface displays a Scala REPL session where a Spark SQL query is executed. The query is: `select age, marital, count(*) as number from datanewtable where y='yes' group by age,marital order by number desc`. The output is a DataFrame with 20 rows, showing columns `age`, `marital`, and `number`. The data is sorted by `number` in descending order. The first 20 rows are displayed, with the last row being `31` `married` `80`.

```
scala> val ageandmaritaldata = spark.sql("select age, marital, count(*) as number from datanewtable where y='yes' group by age,marital order by number desc")
ageandmaritaldata: org.apache.spark.sql.DataFrame = [age: bigint, marital: string ... 1 more field]

scala> ageandmaritaldata.show()
+---+-----+
|age|marital|number|
+---+-----+
| 30|single| 151|
| 28|single| 138|
| 29|single| 133|
| 32|single| 124|
| 26|single| 121|
| 34|married| 118|
| 31|single| 111|
| 27|single| 110|
| 35|married| 101|
| 36|married| 100|
| 25|single| 99|
| 37|married| 98|
| 33|married| 97|
| 33|single| 97|
| 32|married| 87|
| 39|married| 87|
| 38|married| 86|
| 35|single| 84|
| 47|married| 83|
| 31|married| 80|
+---+-----+
only showing top 20 rows
```

Code

```
val agedata = spark.udf.register("agedata",(age:Int) => {

if (age < 20)

"Teen"

else if (age > 20 && age <= 32)

"Young"

else if (age > 33 && age <= 55)

"Middle Aged"

else

"old"

})
```

Output

```
scala> val banknewDF = bank_people_data.withColumn("age", agedata(bank_people_data("age")))
banknewDF: org.apache.spark.sql.DataFrame = [age: string, balance: bigint ... 15 more fields]
```

```
scala> banknewDF.show()
```

| | age | balance | campaign | contact | day | default | duration | education | housing | job | loan | marital | month | pdays | poutcome | previous | y |
|-------------|-------|---------|----------|---------|-----|---------|----------|-----------|---------|--------------|------|----------|-------|-------|----------|----------|----|
| Middle Aged | old | 2143 | 1 | unknown | 5 | no | 261 | tertiary | yes | management | no | married | may | -1 | unknown | 0 | no |
| Middle Aged | old | 29 | 1 | unknown | 5 | no | 151 | secondary | yes | technician | no | single | may | -1 | unknown | 0 | no |
| Middle Aged | old | 2 | 1 | unknown | 5 | no | 76 | secondary | yes | entrepreneur | yes | married | may | -1 | unknown | 0 | no |
| Middle Aged | old | 1506 | 1 | unknown | 5 | no | 92 | unknown | yes | blue-collar | no | married | may | -1 | unknown | 0 | no |
| Middle Aged | old | 1 | 1 | unknown | 5 | no | 198 | unknown | no | unknown | no | single | may | -1 | unknown | 0 | no |
| Middle Aged | Young | 231 | 1 | unknown | 5 | no | 139 | tertiary | yes | management | no | married | may | -1 | unknown | 0 | no |
| Middle Aged | Young | 447 | 1 | unknown | 5 | no | 217 | tertiary | yes | management | yes | single | may | -1 | unknown | 0 | no |
| Middle Aged | old | 2 | 1 | unknown | 5 | yes | 380 | tertiary | yes | entrepreneur | no | divorced | may | -1 | unknown | 0 | no |
| Middle Aged | old | 121 | 1 | unknown | 5 | no | 50 | primary | yes | retired | no | married | may | -1 | unknown | 0 | no |
| Middle Aged | old | 593 | 1 | unknown | 5 | no | 55 | secondary | yes | technician | no | single | may | -1 | unknown | 0 | no |
| Middle Aged | Young | 270 | 1 | unknown | 5 | no | 222 | secondary | yes | admin. | no | divorced | may | -1 | unknown | 0 | no |
| Middle Aged | Young | 390 | 1 | unknown | 5 | no | 137 | secondary | yes | admin. | no | single | may | -1 | unknown | 0 | no |
| Middle Aged | old | 6 | 1 | unknown | 5 | no | 517 | secondary | yes | technician | no | married | may | -1 | unknown | 0 | no |
| Middle Aged | old | 71 | 1 | unknown | 5 | no | 71 | unknown | yes | technician | no | married | may | -1 | unknown | 0 | no |
| Middle Aged | old | 162 | 1 | unknown | 5 | no | 174 | secondary | yes | services | no | married | may | -1 | unknown | 0 | no |
| Middle Aged | old | 229 | 1 | unknown | 5 | no | 353 | primary | yes | retired | no | married | may | -1 | unknown | 0 | no |
| Middle Aged | old | 13 | 1 | unknown | 5 | no | 98 | unknown | yes | admin. | no | single | may | -1 | unknown | 0 | no |
| Middle Aged | old | 52 | 1 | unknown | 5 | no | 38 | primary | yes | blue-collar | no | married | may | -1 | unknown | 0 | no |
| Middle Aged | old | 60 | 1 | unknown | 5 | no | 219 | primary | yes | retired | no | married | may | -1 | unknown | 0 | no |
| Middle Aged | old | 0 | 1 | unknown | 5 | no | 54 | secondary | yes | services | no | married | may | -1 | unknown | 0 | no |

only showing top 20 rows

Code

```
val targetage = spark.sql("select age, count(*) as number from banknewtable where y='yes' group by age order by number desc")
```

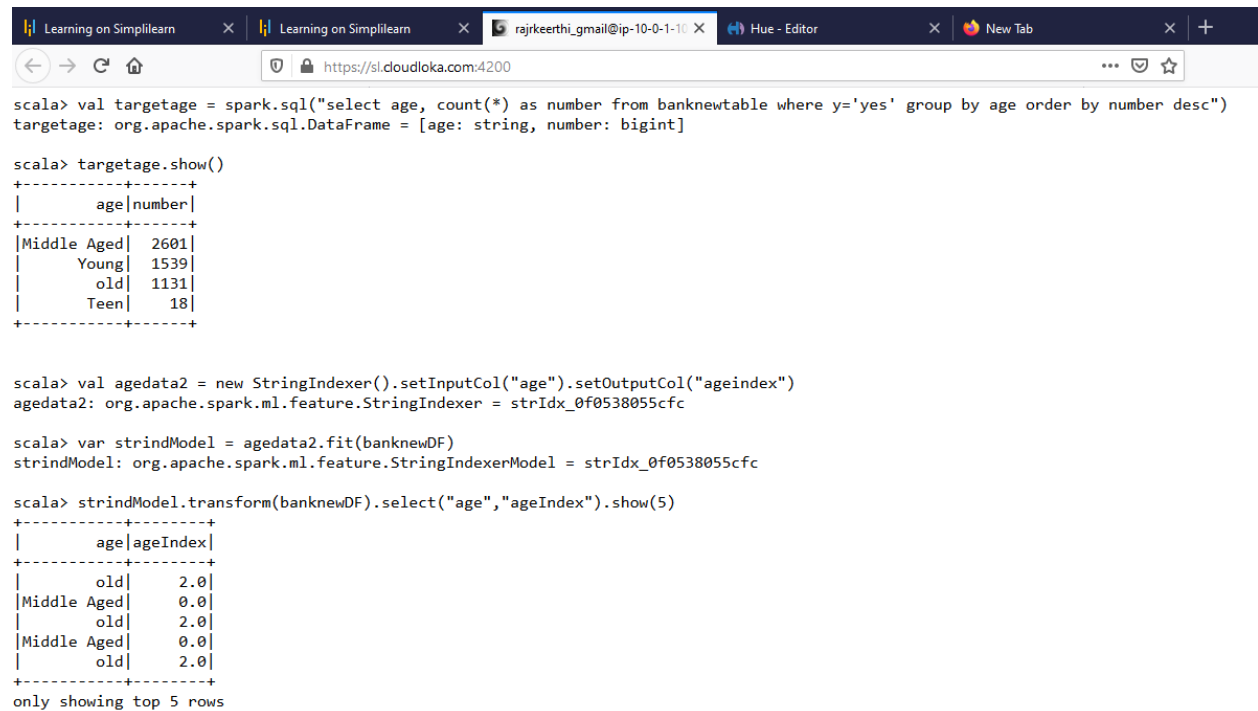
```
targetage.show()
```

```
val agedata2 = new StringIndexer().setInputCol("age").setOutputCol("ageindex")
```

```
var strindModel = agedata2.fit(banknewDF)
```

```
strindModel.transform(banknewDF).select("age","ageIndex").show(5)
```

Output



```
scala> val targetage = spark.sql("select age, count(*) as number from banknewtable where y='yes' group by age order by number desc")
targetage: org.apache.spark.sql.DataFrame = [age: string, number: bigint]

scala> targetage.show()
+-----+-----+
|          age|number|
+-----+-----+
|Middle Aged|   2601|
|      Young|   1539|
|         old|   1131|
|         Teen|    18|
+-----+-----+

scala> val agedata2 = new StringIndexer().setInputCol("age").setOutputCol("ageindex")
agedata2: org.apache.spark.ml.feature.StringIndexer = strIdx_0f0538055cfc

scala> var strindModel = agedata2.fit(banknewDF)
strindModel: org.apache.spark.ml.feature.StringIndexerModel = strIdx_0f0538055cfc

scala> strindModel.transform(banknewDF).select("age","ageIndex").show(5)
+-----+-----+
|          age|ageIndex|
+-----+-----+
|         old|        2.0|
|Middle Aged|        0.0|
|         old|        2.0|
|Middle Aged|        0.0|
|         old|        2.0|
+-----+-----+
only showing top 5 rows
```